A direct proof of the confluence of combinatory strong reduction

René David

A direct proof of the confluence of combinatory strong reduction

René David
Université de Savoie, Campus Scientifique
73376 Le Bourget du Lac, France.
Email : david @univ-savoie.fr

May 15, 2009

Abstract

I give a proof of the confluence of combinatory strong reduction that does not use the one of λ-calculus. I also give simple and direct proofs of a standardization theorem for this reduction and the strong normalization of simply typed terms.

1 Introduction

Combinatory Logic (see [2], [3]) is a first order language that simulates the λ-calculus without using bounded variables. But, at present, the known proofs of confluence are all based on the confluence of the λ-calculus which has to be proved before and thus Combinatory Logic is not a self-contained theory. The question of getting a direct proof of this confluence was raised long ago in [2] and appears in the TLCA list of open problems. I give here such a proof.

The paper is organized as follows. Section 2 gives the main definitions of Combinatory Logic, states the theorem and the idea of the proof. Section 3 gives the proof of the confluence of an auxiliary system. Section 4 gives the equivalence of the two systems and deduce the confluence of the original one. Section 5 gives a standardization theorem and section 6 gives a direct proof of strong normalization for simply typed terms. Finally, I conclude in section 7 with some remarks.

2 The idea of the proof of confluence

2.1 Combinatory Logic

Definition 1 *The set C of combinators is defined by the following grammar (where x denotes a variable)*

$$C := x \;\mid\; K \;\mid\; S \;\mid\; I \;\mid (C\;C)$$

In the literature, the objects determined by this grammar are usually called CL-terms and the word combinator is given for *closed* CL-terms. However since, in section 3, the word term will be used for something slightly different, I prefer to keep the word combinator here.

Definition 2 *For $u \in C$, the term $[x]u$ is defined, by induction on u, by the following rules*

1. $[x]u = Ku$ *if* $x \notin u$
2. $[x]x = I$
3. $[x](u\ x) = u$ *if* $x \notin u$
4. $[x](u\ v) = (S\ [x]u\ [x]v)$ *if none of the previous rules apply.*

Definition 3 *The reduction on combinators is the closure by contexts of the following rules.*

1. $(K\ u\ v) \succ u$ $\qquad (S\ u\ v\ w) \succ (u\ w\ (v\ w))$ $\qquad (I\ u) \succ u$
2. $[x]u \succ [x]v$ $\qquad$ *if* $u \succ v$

I recall here usual notions about reductions.

Definition 4 *Let* $\rightarrow$ *be a notion of reduction.*

- *As usual,* $\rightarrow^*$ *denotes the reflexive and transitive closure of* $\rightarrow$.
- *The reduction* $\rightarrow$ *is locally confluent if, for any term* u*, the following holds. If* $u \rightarrow u_1$ *and* $u \rightarrow u_2$*, then* $u_1 \rightarrow^* u_3$ *and* $u_2 \rightarrow^* u_3$ *for some* u_3.
- *The reduction* $\rightarrow$ *commutes with the reduction* $\rightarrow_1$ *if, for any term* u*, the following holds. If* $u \rightarrow^* u_1$ *and* $u \rightarrow_1^* u_2$ *then* $u_1 \rightarrow_1^* u_3$*,* $u_2 \rightarrow^* u_3$ *for some* u_3
- *The reduction* $\rightarrow$ *is confluent if it commutes with itself.*
- *A term* u *is strongly normalizing (denoted as* $u \in SN$*) if there is no infinite reduction of* u.

Remark 5 *Rule (2) of definition 3 is fundamental to have the equivalence of combinatory logic (denote as LC) and* λ*-calculus (denoted as* Λ*) in the following sense. Let* H *be the translation between* Λ *and* LC *defined by*

$$H(x) = x \qquad H((u_1\ u_2)) = (H(u_1)\ H(u_2)) \qquad H(\lambda x.u) = [x]H(u)$$

Without rule (2), the compatibility property between Λ *and* LC *(i.e. if* t *reduces to* t'*, then* $H(t)$ *reduces to* $H(t')$*) would not be true. This is because the reduction in* LC *will not allow a reduction below a* λ*. For example, let* $t = \lambda x.(\lambda y.x\ x)$*. Then* $H(t) = [x](K\ x\ x) = (S\ K\ I)$ *is normal whereas* t *is not.*

Note that without rule (2) of definition 3 (this reduction is then called weak reduction), the confluence would be trivially proved by the method of parallel reductions.

Remark 6 *The confluence of the reduction* $\succ$ *depends on the good interaction between rule (2) of definition 3 and clause (3) of definition 2 (which corresponds, intuitively, to the* η*-equality of the* λ*-calculus). In fact, the confluence of* $\succ$ *would not be true if clause (3) of definition 2 had been omitted. The reason is the following. Let* u *and* v *be two combinators. Assume variable* x *occurs in* u *but not in* v *and* u *reduces to* u' *for some* u' *that does not contain* x *(for example* $u = K\ y\ x$*). Then, by applying rule (2) of definition 3, we have*

(1) $\qquad [x](u\ v) = (S\ [x]u\ (K\ v)) \succ (S\ [x]u'\ (K\ v)) = (S\ (K\ u')\ (K\ v))$

and

(2) $\qquad [x](u\ v) \succ [x](u'\ v) = (K\ (u'\ v))$

- *Without clause (3) of definition 2 the term* $(S\ (K\ u')\ (K\ v))$ *is not of the form* $[x]w$*, so that the two terms* $(K\ (u'\ v))$ *and* $(S\ (K\ u')\ (K\ v))$ *are not reducible to a common term.*

- *With clause (3) of definition 2 the term* $(S\ (K\ u')\ (K\ v))$ *is of the form* $[x]w$*.*

$$(S\ (K\ u')\ (K\ v)) = [x](S\ (K\ u')\ (K\ v)\ x)$$

from which

$$[x](S\ (K\ u')\ (K\ v)\ x) \succ [x](K\ u'\ x\ (K\ v\ x)) \succ^* [x](u'\ v) = (K\ (u'\ v))$$

Thus the two terms $(K\ (u'\ v))$ *and* $(S\ (K\ u')\ (K\ v))$ *are reducible to a common term.*

The main result of this paper is the following theorem.

Theorem 7 *The reduction* $\succ$ *on combinators is confluent.*

2.2 The idea of the proof

I want to prove the confluence by using the same method as in [1] i.e. by proving first a theorem on finiteness of developments. Then, by this theorem, Newman's Lemma and the local confluence of the developments we get the confluence of developments. Then it remains to show that the reduction itself is the transitive closure of the developments.

But the given system is quite hard to study because it is difficult to mark the redexes and thus to give a precise definition for a theorem on finiteness of developments. This is also because the form of a term does not determine easily its redexes. The main technical reason is the following. We should think that any reduct of $[x]u$ would have the form $[x]u'$ for some reduct u' of u. But this property, which is trivial in the λ-calculus, is not true here. Here is an example. Let u, v be combinators, x be a variable that occurs both in u and v and let $t = [x](u\ v) = (S\ [x]u\ [x]v)$. Then, it is easy to check that $t = ([y][x](u\ (y\ x))\ [x]v)$. Now if $u = (S\ u_1\ u_2)$ then t reduces to $t' = ([y][x](u_1\ (y\ x)\ (u_2\ (y\ x)))\ [x]v)$ and it is easy to check that t' cannot be written as $[x]w$ for some reduct w of $(u\ v)$. Note that, in the λ-calculus, the corresponding equality i.e. $\lambda x.(u\ v) = (\lambda yx.(u\ (y\ x))\ \lambda x.v)$ needs β-reductions and not only η-reductions whereas in Combinatory Logic it only comes from the η-rule.

Thus I will first prove the confluence of an auxiliary system. This system will be shown to be equivalent to the other one in the sense that the symmetric and transitive closure of both systems are the same. Then I will deduce the confluence of the first system from the one of the second.

The auxiliary system treats separately the reductions that, intuitively, corresponds in the λ-calculus to β and η. To prove the confluence of this system, I prove the confluence of β. This is done, as mentioned above, by proving a theorem on finiteness of developments. Note that the fact that the reduction is the transitive closure of developments (which is trivial in the λ-calculus) is not so easy here. I deduce the confluence of the whole system (intuitively β and η) by another commutation lemma.

Lemma 8 (Newman's Lemma) *Let* $\rightarrow$ *be a notion of reduction that is locally confluent and strongly normalizing. Then* $\rightarrow$ *is confluent.*

3 An auxiliary system

To define this new system, I first remove the η-equality in the definition of the abstraction.

Definition 9 *1.* $\lambda x.u = (K\ u)$ *if* $x \notin u$

2. $\lambda x.x = I$

3. $\lambda x.(u\ v) = (S\ \lambda x.u\ \lambda x.v)$ *if none of the previous rules apply.*

and I add new reduction rules. In Definition 10 below rule (2) is necessary to have confluence. Rule (3) corresponds to the η-reduction and is necessary to have the equivalence with the other system.

Definition 10 *1.* $(K\ u\ v) \to u \qquad (S\ u\ v\ w) \to (u\ w\ (v\ w)) \qquad (I\ u) \to u$

2. $(S\ (K\ u)\ (K\ v)) \to (K\ (u\ v))$

3. $(S\ (K\ u)\ I) \to u$

4. $\lambda x.u \to \lambda x.v \qquad$ *if* $u \to v$

It is important to note that the two reductions $\succ$ and $\to$ are not the same i.e. there are combinators such that $u \to^* v$ for some v but u does not reduce to v by $\succ$ and, similarly, there are combinators such that $u \succ^* v$ for some v but u does not reduce to v by $\to$. Here are examples. Let $u = [y][x](S\ x\ x\ (y\ x))$. Then $u \succ [y][x](x\ (y\ x)\ (x\ (y\ x)))$ and it is easy to check that u is normal for $\to$. Let $u_1 = \lambda x.(S\ y\ x\ x) \to \lambda x.(y\ x\ (x\ x)) = v$ and it is not too difficult to check that u does not reduce to v by $\succ$.

Although the two reductions $\succ$ and $\to$ are not the same, we now show that they give the same equations on combinators. I denote by $\equiv$ the equivalence relation induced by $\succ$ i.e. $u \equiv v$ iff there is a sequence $u_0, ..., u_n$ of combinators such that $u_0 = u$, $u_n = v$ and, for each i, either $u_i \succ u_{i+1}$ or $u_{i+1} \succ u_i$. The equivalence induced by $\to$ will be denoted by $\approx$.

Lemma 11 *1. For each* u, v, $(S\ (K\ u)\ I) \succ^* u$ *and* $(S\ (K\ u)\ (K\ v)) \succ^* (K\ (u\ v))$

2. For each u, $\lambda x.u \to^* [x]u$ *and* $\lambda x.u \succ^* [x]u$.

Proof

1. Let x be a fresh variable. Then, $(S\ (K\ u)\ I) = [x](S\ (K\ u)\ I\ x) \succ [x](K\ u\ x\ (I\ x)) \succ^* [x](u\ x) = u$ and $(S\ (K\ u)\ (K\ v)) = [x](S\ (K\ u)\ (K\ v)\ x) \succ [x](K\ u\ x\ (K\ v\ x)) \succ^* [x](u\ v) = (K\ (u\ v)))$.

2. This follows immediately from the first point.

□

Theorem 12 *Let* u, v *be combinators. Then* $u \equiv v$ *iff* $u \approx v$.

Proof It is enough to show that if $u \succ v$ then $u \approx v$ and if $u \to v$ then $u \equiv v$. Each point is proved by induction on the level of the reduction. The result is trivial for the level 0. Assume then that the level is at least 1. For the first direction, I have to show that, if $u \succ v$ then $[x]u \approx [x]v$. By the *IH* we know that $u \approx v$ and it is thus enough to show that, if $u \to v$, then $[x]u \approx [x]v$. By the previous lemma, we have $\lambda x.u \to [x]u$ and, since $\lambda x.u \to \lambda x.v \to [x]v$, we are done. For the other direction, we have to prove that, if $u \succ v$, then $\lambda x.u \equiv \lambda x.v$. This is because $\lambda x.u \succ [x]u \succ [x]v$ and $\lambda x.v \succ [x]v$. □

Theorem 13 *The reduction $\rightarrow$ on combinators is confluent.*

As mentioned before, to prove this theorem I first prove the confluence of the system where the η-reduction (i.e. rule (3) of definition 10) has been removed. The theorem on finiteness of developments of this system can be formalized as theorem 33 below. I need some new definitions.

3.1 Some definitions

Definition 14 *Let V be an infinite set of variables.*

- *Let $A = V \cup \{S_i \ / \ i = 0, 1, 2, 3\} \cup \{K_i \ / \ i = 0, 1\} \cup \{I_i \ / \ i = 0, 1\}$. The elements of A will be called atoms.*
- *The set of terms is defined by the following grammar*

$$T := A \mid \quad (T \ T)$$

- *The size of a term (denoted as $size(t)$) is defined by the following rules: for $\alpha \in A$, $size(\alpha) = 1$ and $size((u \ v)) = size(u) + size(v) + 1$.*

The meaning of the indices on S, K, I is the following. First, I want to mark the redexes that are allowed to be reduced. I do this by simply indexing the letters S, K, I. The index 0 means that the symbol is not marked (i.e. we are not allowed to reduce the corresponding redex), the index 1 means that the redex is allowed.

I also want to indicate whether or not a combinator S, K, I is the first symbol of a term of the form $\lambda x.u$ for which I want to reduce in u. Actually, for K, I there is nothing to do because a variable has no redex and, since $\lambda x.u = (K \ u)$ when x does not occur in u, the redexes in u are, in fact, already visible at the top level. But for S this will be useful and I need thus 4 indices.

- S_0 is an S that is neither marked nor introduced by a λ,
- S_1 is an S that is marked but not introduced by a λ,
- S_2 is an S that is not marked but introduced by a λ
- S_3 is an S that is marked and introduced by a λ.

Definition 15 *Let u be a term and x be a variable. I define, for $i = 0, 1$ the set of terms (denoted as $\lambda_i x.t$) by the following rules.*

1. *if $t = x$, $\lambda_i x.t = \{I_i\}$*
2. *if $t \neq x$ is an atom, $\lambda_i x.t = \{(K_i \ t)\}$*
3. *if $t = (u \ v)$ and $x \notin t$, $\lambda_i x.t = \{(K_i \ t)\} \cup \{(S_{i+2} \ u' \ v') \mid u' \in \lambda_i x.u, v' \in \lambda_i x.v\}$*
4. *if $t = (u \ v)$ and $x \in t$, $\lambda_i x.t = \{(S_{i+2} \ u' \ v') \mid u' \in \lambda_i x.u, v' \in \lambda_i x.v\}$.*

The reason of this unusual definition and, in particular, the fact that $\lambda_i x.t$ represents a set of terms instead of a single term, is the following. It will be useful to ensure that the set of terms of the form $\lambda x.u$ is closed by reduction. But this is not true if the abstraction is defined by the rules of definition 9.

Here is an example. Let u and v be two combinators. Assume variable x occurs in u but not in v and u reduces to u' for some u' that does not contain x. As shown in points (1) and (2) of Remark 6, $\lambda x.(u \ v)$ reduces to $(S \ (K \ u') \ (K \ v))$ and

$(K\ (u'\ v))$. Allowing, in such a case, both $(K\ (u'\ v))$ and $(S\ (K\ u')\ (K\ v))$ to be in $\lambda x.(u'\ v)$ will repair this problem.

The given definition is then an indexed version of this idea. The index 1 (resp. 0) will mean that the S, K, I introduced by the definition are marked (resp. are not marked) and thus allow a redex to be reduced. Note that the $i+2$ indexing S means (depending whether $i=1$ or $i=0$) that S comes from a λ and is (or is not) marked.

Definition 16 *The reduction (denoted as $t \triangleright t'$) on terms is the closure by contexts of the following rules*

1. *(a) For $i = 1, 3$ $(S_i\ u\ v\ w) \triangleright (u\ w\ (v\ w))$*
   (b) $(K_1\ u\ v) \triangleright u$ and $(I_1\ u) \triangleright u$
   (c) For $i = 0, 1$ $(S_{i+2}\ (K_i\ u)\ (K_i\ v)) \triangleright (K_i\ (u\ v))$
   (d) For $i = 0, 1$, if $u \triangleright v$, $t \in \lambda_i x.u$ and $t' \in \lambda_i x.v$, then $t \triangleright t'$
2. *The level of a reduction (denoted as $lvl(t \triangleright t')$) is defined as follows.*
   - *If $t \triangleright t'$ by using rule (a),(b) or (c), the level is 0.*
   - *$t \triangleright t'$ by using rule (d), the level is $lvl(u \triangleright v) + 1$.*

Remarks and examples

These rules correspond to the indexed version of the rules (1, 2, 4) of definition 10 combined with the fact that $\lambda x.u$ now is a set of terms.

For example, if x does not occur in $(u\ v)$ and $u \triangleright u'$, since $(K_i\ (u\ v)) \in \lambda_i x.(u\ v)$ and $(S_{i+2}\ (K_i\ u')\ (K_i\ v)) \in \lambda_i x.(u'\ v)$ we have $(K_i\ (u\ v)) \triangleright (S_{i+2}\ (K_i\ u')\ (K_i\ v))$. Note that $(K\ (u\ v))$ does not reduce to $(S\ (K\ u')\ (K\ v))$ by the rules of Definition 10.

3.2 Fair terms

We will show the confluence of $\triangleright$ not of the entire set of terms but on some subset (the set of fair terms) that we now define. This is because we need a set that is closed by reduction (see Lemma 31).

Notation 17
- *Let E be a set of terms and $\overrightarrow{u}$ be a sequence of terms (resp. f be function into terms). I will write $\overrightarrow{u} \in E$ (resp. $f \in E$) to express the fact that each term of the sequence $\overrightarrow{u}$ (resp. in the image of f) is in E.*
- *Let $\overrightarrow{u}$ be a finite (possibly empty) sequence of terms and v be a term. I denote by $(v\ \overrightarrow{u})$ the term $(v\ u_1\ ...\ u_n)$ where $\overrightarrow{u} = u_1, ..., u_n$.*

Definition 18
- *An address is a finite list of elements of the set $\{l, r\}$.*
- *The empty list will be denoted by ε and $[a :: l]$ (resp. $[l :: a]$) will denote the list obtained from a by adding l at the end (resp. at the beginning) of a and similarly for r.*
- *If a, a' are addresses, I will denote by $a < a'$ the fact that a is an initial segment of a'.*
- *Let u be a term. I will denote by u_a the sub-term of u at the address a. More precisely, u_a is defined by the following rules: $u_\varepsilon = u$, $(u\ v)_{[l::a]} = u_a$ and $(u\ v)_{[r::a]} = v_a$.*

Definition 19
- *Let u be a term and f be a function from a set E of addresses in u into terms. I say that f is adequate for u (I will also say (u, f) is adequate) if there are no addresses a, a' in E such that $a < a'$.*

- *Let (u, f) be adequate and x be a variable. Then $\phi_x(u, f)$ is a term obtained by replacing in u, for each $a \in dom(f)$, the term at address a by $(w_a\ f(a))$ for some $w_a \in \lambda_1 x.u_a$.*

- *Let u be a term, $x_1, ..., x_n$ (resp. $f_1, ..., f_n$) be a sequence (possibly empty) of variables (resp. of functions). The term $\phi_{x_1}(\phi_{x_2}(...(\phi_{x_n}(u, f_n), f_{n-1})...)f_1)$ will be denoted by $\phi(u, \overrightarrow{x}, \overrightarrow{f})$ or simply $\phi(u)$ if we do not need to mention explicitly $\overrightarrow{x}, \overrightarrow{f}$ or if they are clear from the context.*

Comments and examples

A typical term of the form $\phi_x(u, f)$ is obtained as follows. Let $t = (\lambda_1 x.u\ v)$. First reduce the head redex of t (this intuitively means: do the β-reduction and introduce a kind of explicit substitution $[x := v]$) and then propagate (not necessarily completely) this substitution inside u (this intuitively means do some S, K, I reductions at the top level), possibly doing some (different) reductions in the (different) occurrences of v. The term obtained in this way is a typical term of the form $\phi_x(u, f)$. Here is an example.

Let $u = (y\ x\ x)$, v, v' be combinators and let f be such that $f([l]) = v$ and $f([r]) = v'$. Then $\phi_x(u, f) = (S_3\ (K_1\ y)\ I_1\ v\ (I_1\ v'))$. Remark that, if $v \triangleright v'$, we have $(\lambda_1 x.u\ v) \triangleright \phi_x(u, f)$.

Note that, even if we only need $\phi_x(u, f)$ in case the terms in the image of f are reducts of a single term, we do not ask this property in the definition.

Finally note that, in the same way that x does not occur in $\lambda_i x.u$, it does not occur in $\phi_x(u, f)$. This implies that, as usual, when we substitute a variable y by some term v in a term of the form $\lambda_i x.u$ or $\phi_x(u, f)$ we may assume (by possibly renaming x with a fresh name) that x does not occur in v, avoiding then its capture.

Definition 20 *The set F of fair terms is defined by the following grammar.*

1. *x, S_0, K_0, I_0 are fair*
2. *If u, v are fair then so is $(u\ v)$.*
3. *If u is fair and $t \in \lambda_0 x.u$ then so is t.*
4. *If v_1, v_2, v_3 are fair, then so are $(S_1\ v_1\ v_2\ v_3)$, $(K_1\ v_1\ v_2)$ and $(I_1\ v_1)$*
5. *If x is a variable, $u, f \in F$ and (u, f) is adequate, then $\phi_x(u, f)$ is fair.*

Fair terms are thus combinators where we have marked the redexes that are allowed to be reduced. The terms of the form $\phi_x(u, f)$ are introduced for the following reason. If $t = (w\ v)$ for some $w \in \lambda_1 x.u$, I may want to reduce both a redex in u and t as a redex. Thus the set of fair terms must be closed by the following rule: (6) If u, v are fair then so is $t = (w\ v)$ for $w \in \lambda_1 x.u$. But, if I had defined fair terms by rules 1, 2, 3, 4 and 6, then F will not be closed by reduction because, if $w \in \lambda_1 x.u$, the reduct of $t = (w\ v)$ will not necessarily be fair. The reason is the following. Let $u = (u_1\ u_2)$ be such that u is fair but u_1 is not (for example $u_1 = (K_1\ y), u_2 = y$). Then $v = (\lambda_1 x.u\ z)$ is fair. But $v \triangleright v' = (\lambda_1 x.u_1\ z)\ (\lambda_1 x.u_2\ z)$ and v' may not be fair since u_1 is not.

Definition 21 *Let u be fair. I denote by $nb(u)$ the number of rules that have been used to prove that u is fair.*

3.3 Some properties of fair terms

Lemma 22 *The set of fair terms is closed by substitutions.*

Proof By an immediate induction on $nb(u)$. Use the fact that, if $t \in \lambda_i x.u$, then $\sigma(t) \in \lambda_i x.\sigma(u)$. □

Lemma 23 *Let $t = (\alpha\ \overrightarrow{u})$ be fair where α is an atom.*

1. *If α is S_2, then $lg(\overrightarrow{u}) \geq 2$. If α is S_1 or S_3, then $lg(\overrightarrow{u}) \geq 3$.*
2. *If α is K_1, then $lg(\overrightarrow{u}) \geq 2$. If α is I_1, then $lg(\overrightarrow{u}) \geq 1$.*

Proof By induction on $nb(t)$. I only look at the cases with S. The other ones are similar.

- If the last rule that has been used to prove $t \in F$ is (2) of definition 20, the result follows immediately from the *IH*. If it is rule (4) the result is trivial.
- If it is rule (3). If $\alpha = S_2$, the result is also trivial. The other cases are impossible.
- If it is rule (5) and $(\alpha\ \overrightarrow{u}) = \phi_y(v, f)$. Let a be the leftmost address in $dom(f)$. For $\alpha = S_1$ (resp. $\alpha = S_2$) we may not have $a = [l, l, ..., l]$ since this will imply that t begins with S_3. Thus $v = (S_1\ \overrightarrow{w})$ (resp. $v = (S_2\ \overrightarrow{w})$) and the result follows from the *IH*. For $\alpha = S_3$, if the leftmost address is not of the form $[l, l, ..., l]$ the result is as before. Otherwise, this implies that $t = (w_a\ f(a)\ \overrightarrow{s})$ for some $w_a \in \lambda_1 y.v_a$ and some $\overrightarrow{s}$ and the result is trivial.

$\square$

Lemma 24 *Let u, u' be terms, $t \in \lambda_i y.u$ and $t' \in \lambda_j x.u'$. Assume t is a sub-term of t'. Then, either t is a sub-term of u' or $i = j$, $x = y$ and u is a sub-term of u'.*

Proof By induction on u'. $\square$

Lemma 25

- *Let $t = (\alpha\ \overrightarrow{u}) \in F$ where $\alpha \in V \cup \{S_i, K_i, I_i\ /i = 0, 1\}$. Then, $\overrightarrow{u} \in F$.*
- *If $t = (S_2\ \overrightarrow{u}) \in F$, then $t = \phi((r\ \overrightarrow{w}))$ for some $r \in \lambda_0 y.v$ and some $v, \overrightarrow{w} \in F$.*

Proof By induction on $nb(t)$, essentially as in lemma 23. $\square$

3.4 Some properties of reduction

Lemma 26 *Let u_1, u_2 be fair and assume $t = (u_1\ u_2) \triangleright t'$. Then $t' = (u_1'\ u_2)$ or $t' = (u_1\ u_2')$ where $u_i \triangleright u_i'$.*

Proof It is enough to show that there is no possible interaction between u_1 and u_2. Such an interaction could occur in the following cases.
- $lvl(t \triangleright t') = 0$. This is impossible because, by Lemma 23, all the arguments of the indexed S, K or I of such a redex must be in u_1.
- $lvl(t \triangleright t') > 0$ and, for example, $t \in \lambda_0 x.v$ and $t' \in \lambda_0 x.v'$ for some $v \triangleright v'$. This could occur if $u_1 = (S_2\ w_1)$ for some $w_1 \in \lambda_0 x.t_1$, $u_2 \in \lambda_0 x.t_2$ and $v = (t_1\ t_2)$. But this is again impossible by Lemma 23. $\square$

Lemma 27 *Let u_1, u_2, u_3 be terms.*

- *Assume $t = (I_1\ u_1) \triangleright t'$. Then either $t' = u_1$ or $t' = (I_1\ u_1')$ for $u_1 \triangleright u_1'$.*
- *Assume $t = (K_1\ u_1\ u_2) \triangleright t'$. Then either $t' = u_1$ or $t' = (K_1\ u_1'\ u_2)$ or $t' = (K_1\ u_1\ u_2')$ for $u_i \triangleright u_i'$.*
- *Assume $t = (S_1\ u_1\ u_2\ u_3) \triangleright t'$. Then either $t' = (u_1\ u_3\ (u_2\ u_3))$ or $t' = (S_1\ u_1'\ u_2'\ u_3')$ where $u_i \triangleright u_i'$ for a unique i and $u_j' = u_j$ for $j \neq i$.*

Proof It is enough to show that the mentioned reductions are the only possibilities. I only look at the last case since the other ones are similar.

If $lvl(t \triangleright t') = 0$, the result is trivial. Otherwise, this means that there is a sub-term of $t \in \lambda_i x.v$ which reduces to a term in $\lambda_i x.v'$ for $v \triangleright v'$. But, this sub-term has to be a sub-term of some u_j because, otherwise (by Definition 15) we will have S_2 or S_3 instead of S_1, and the result follows immediately. □

Lemma 28 *Assume $t \in \lambda_0 x.u$ and $t \triangleright t'$. Then either $t' \in \lambda_0 x.u$ and $size(t') < size(t)$ or $t' \in \lambda_0 x.u'$ for some u' such that $u \triangleright u'$.*

Proof If $lvl(t \triangleright t') = 0$, the reduction cannot use (the closure by context of) rule (a) in Definition 16. This is because, since $t \in \lambda_0 x.u$, the index of S in the reduced redex cannot be 1 or 3 and thus the result is clear. Otherwise, this follows easily from Lemma 24. □

Lemma 29 *Assume $\phi(u, \overrightarrow{y}, \overrightarrow{f}) \in \lambda_0 x.v$. Then $u \in \lambda_0 x.w$ for some w such that $\phi(w, \overrightarrow{y}, \overrightarrow{f}) = v$.*

Proof By an immediate induction on the length of the sequence $\overrightarrow{y}$ it is enough to prove the result for $\phi_y(u, f)$. This is proved by induction on v. I only consider the case $v = (v_1\ v_2)$ and $\phi_y(u, f) = (S_2\ r_1\ r_2)$ where $r_j \in \lambda_0 x.v_j$ (the other cases are similar). The leftmost address in $dom(f)$ cannot be $[l, l, ..., l]$ because, otherwise, $\phi_y(u, f)$ will begin with S_3. Thus u is an application and $\phi_y(u, f) = (S_2\ \phi_y(u_1, f_1)\ \phi_y(u_2, f_2))$ where $u = (u_1\ u_2)$. Thus $\phi_y(u_i, f_i) \in \lambda_0 x.v_i$ and we conclude by the *IH*. □

Lemma 30 *Let $u, f \in F$ be such that (u, f) is adequate. Then a redex in $t = \phi_x(u, f)$ is either in u or in some $f(a)$ or is $(w_a\ f(a))$ for some a and some $w_a \in \lambda_1 x.u_a$. Thus, if $t \triangleright t'$, one of the following cases holds.*

- *$t' = \phi_x(u', f')$ for some u', f' such that $u \triangleright u'$*
- *$t' = \phi_x(u, f')$ where $f \triangleright f'$*
- *t' is obtained from t by reducing the redex $(w_a\ f(a))$ for some $a \in dom(f)$ and some $w_a \in \lambda_1 x.u_a$. Then, $t' = \phi_x(u', f')$ and*
  - *If $u_a = x$, then u' is u where the occurrence of x at the address a has been replaced by $f(a)$ and $dom(f') = dom(f) - \{a\}$.*
  - *If $x \notin u_a$, then $u' = u$ and $dom(f') = dom(f) - \{a\}$.*
  - *If $u_a = (v_1\ v_2)$ then $u' = u$, $dom(f') = dom(f) - \{a\} \cup \{[a :: l], [a :: r]\}$, $f'([a :: l]) = f'([a :: r]) = f(a)$ and, for $b \neq a$, $f'(b) = f(b)$.*

Proof By induction on $nb(u)$. The only thing to be shown is that the mentioned cases are the only possible ones. For $lvl(t \triangleright t') = 0$, this follows immediately from the fact that terms of the form $(w_a\ f(a))$ for some $w_a \in \lambda_1 x.u_a$ cannot introduce an interaction since they are redexes. For $lvl(t \triangleright t') > 0$, assume $r \in \lambda_i x.w$ is a sub-term of $\phi_y(u, f)$ and the reduction takes places in w. Then, by Lemma 29, either the reduction is actually in f or $w = \phi_y(v', f')$ for some adequate (v', f') and the result follows from the *IH*. □

Lemma 31

- *The set of fair terms is closed by reduction.*
- *Let u be fair and σ be a fair substitution. Assume $t = \sigma(u) \triangleright t'$, then either $t' = \sigma(u')$ for some $u \triangleright u'$ or $t' = \sigma'(u)$ for some $\sigma \triangleright \sigma'$.*

Proof By induction on $nb(u)$, using Lemmas 26, 27, 28 and 30. □

3.5 Confluence of $\triangleright$ on fair terms

Lemma 32 *Let u be fair and σ be a fair substitution. If $u, \sigma \in SN$, then so is $\sigma(u)$.*

Proof This follows immediately from Lemma 31. $\square$

Theorem 33 *Any fair term t is in SN.*

Proof By induction on $nb(t)$.

- If $t = x, S_0, K_0, I_0$, the result is trivial.
- If $t = (t_1\ t_2)$, then, by the *IH*, $t_1, t_2 \in SN$ and, since $t = \sigma((x\ y))$ where $\sigma(x) = t_1$ and $\sigma(y) = t_2$, the result follows from Lemma 32.
- If $t = (S_1\ t_1\ t_2\ t_3)$, $t = (K_1\ t_1\ t_2)$ or $t = (I_1\ t_1)$ the proof is similar, e.g. $(S_1\ t_1\ t_2\ t_3) = \sigma((S_1\ x_1\ x_2\ x_3)$ where $\sigma(x_i) = t_i$.
- If $t \in \lambda_0 x.v$, the result follows from Lemma 28 and the *IH*.
- Finally, assume $t = \phi_x(u, f)$. Let t' be the term obtained from u by replacing, for each $a \in dom(f)$, u_a by $u_a[x := f(a)]$. It follows from Lemma 32 that $t' \in SN$. But, by Lemma 30, and infinite reduction of t would give an infinite reduction of t' since it is not possible to have infinitely many successive reductions of t of the form of the last case of Lemma 30. Thus t is in SN.

$\square$

Lemma 34 *Let u, v be terms. Then, for $w \in \lambda_1 x.u$, $(w\ v) \triangleright^* u[x := v]$.*

Proof By induction on u. $\square$

Lemma 35 *The reduction $\triangleright$ is locally confluent on fair terms.*

Proof The only critical pairs are the following.

- $t = (w\ u_3)$, $w \in \lambda_1 x.(u_1\ u_2)$, $t \triangleright t_1 = (w_1\ u_3\ (w_2\ u_3))$ for $w_j \in \lambda_1 x.u_j$, and $t \triangleright t_2 = (w'\ u_3)$ for $w' \in \lambda_1 x.v$ and $(u_1\ u_2) \triangleright v$. Both t_1 and t_2 reduces to $v[x := u_3]$.
- $t = (S_{i+2}\ r_1\ r_2) \in \lambda_i x.(u_1\ u_2)$, $x \in u_1$, $x \notin u_2$ (for example), for some $u_1 \triangleright v_1$ such that $x \notin v_1$, $t \triangleright t_1 = (K_i\ (v_1\ u_2))$ and $t \triangleright t_2 = (S_{i+2}\ (K_i\ v_1)\ (K_i\ u_2)))$. But $t_2 \triangleright t_1$.

$\square$

Theorem 36 *The reduction $\triangleright$ is confluent on fair terms.*

Proof By Lemma 8 and 35. $\square$

3.6 Proof of theorem 13

In this section I will still denote by $\triangleright$ the reduction on *combinators* given by rules (1, 2, 4) of definition 10.

Definition 37 • *Let u be a combinator. A labelling of u is a function that associates to each occurrence of S (resp. K, I) in u some S_i (resp. some K_i, I_i).*

- *If L is a labelling of u, I still denote by $L(u)$ the term obtained by replacing in u the symbols S (resp. K, I) by $L(S)$ (resp. $L(K)$, $L(I)$).*
- *Let u be a term. I denote by $\theta(u)$ the combinator defined by the following rules. $\theta(x) = x$, $\theta(S_i) = S$, $\theta(K_i) = K$, $\theta(I_i) = I$ and $\theta((u\ v) = (\theta(u)\ \theta(v))$*
- *Let u be a combinator and L, L' be labelling of u. I say that L' is an extension of L if the following holds.*
  1. *For each S in u,*
     - either $L(S) = L'(S)$
     - or $L(S) = S_0$ and $L'(S) = S_i$ for $i = 1, 2$ or 3
     - or $L(S) = S_2$ or $L(S) = S_1$ and $L'(S) = S_3$.
  2. *For each K in u, $L(K) = L'(K)$ or $L(K) = K_0$ and $L'(K) = K_1$.*
  3. *For each I in u, $L(I) = L'(I)$ or $L(I) = I_0$ and $L'(I) = I_1$.*

A labelling of u is thus a way of marking redexes in u. The function θ consists in un-marking terms to get combinators. Extending a labelling means allowing more redexes to be reduced.

Lemma 38 *Let u be a combinator and L be a labelling of u. If $L(u) \triangleright v$ then $u \triangleright \theta(v)$.*

Proof Immediate. □

Lemma 39 *Assume $t = L(\lambda x.r) \in F$ for some L, r. Then, there is an extension L' of L such that $L'(\lambda x.r) \in \lambda_0 x.v$ for some $v \in F$.*

Proof First note that, for combinators, $\lambda x.r$ represents a single term and thus having written $t = L(\lambda x.r)$ is not a typo !

L' is obtained by iterating the following algorithm.

- If x does not occur in r, choose $L' = L$. Since $t = (L(K)\ L(r))$, by Lemma 23, $L(K)$ must be K_0 and thus, by Lemma 25, $L(r) \in F$.

- If $r = x$, choose $L' = L$. The argument is similar.

- If $r = (r_1\ r_2)$. Then $\lambda x.r = (S\ \lambda x.r_1\ \lambda x.r_2)$. By Lemma 23, $L(S)$ must be either S_0 or S_2.

If $L(S) = S_2$, by Lemma 25, $t \in \phi(\lambda_0 x.v)$ for some $v \in F$ (the term $\phi(u)$ is defined in Definition 19). Thus L satisfies the desired property since, by Lemma 29, t must be in $\lambda_0 x.\phi(v)$.

If $L(S) = S_0$, then, by Lemma 25, $L(\lambda x.r_i) \in F$. Choose $L'(S) = S_2$ and iterate the algorithm with $L(\lambda x.r_j)$ for $j = 1, 2$.

□

Lemma 40 *Let t be a combinator and L be a labelling of t such that $L(t)$ is fair. Assume that $t \triangleright t'$. Then, there is an extension L' of L such that $L'(t)$ is fair and $L'(t) \triangleright v$ for some v such that $\theta(v) = t'$.*

Proof By induction on $nb(L(t))$. Look at the last rule that has been used to show that $L(t)$ is fair.

Rule (3) : a redex in $w \in \lambda_0 x.u$ is either a redex in u (and the result follows immediately from the *IH*) or it is of the form $(S_2\ (K_0\ u_1)\ (K_0\ u_2)) \triangleright (K_0\ (u_1\ u_2))$ and thus already appear in $L(t)$.

Rule (5) : a redex in $\phi_x(u, f)$ is either a redex in u or in some $f(a)$ or a redex already in $L(t)$ and the result follows immediately from the *IH*.

Rule (2) : then $t = (t_1\ t_2)$ and $L(t_1), L(t_2)$ are fair. If the reduced redex is either in t_1 or t_2, the result follows immediately from the *IH*. Otherwise it has been created by the application of t_1 to t_2. I will only look at the cases where the reduced redex starts with some S. The case of K and I are similar and much simpler. For sake of simplicity I will define L' by only mentioning the labels that are changed. We distinguish the different possible redexes.

(a) $t_1 = (S\ u\ v)$ and $t' = (u\ t_2\ (v\ t_2))$.

- If $L(S) = S_0$ then, setting $L'(S) = S_1$ gives the desired properties since, by Lemma 25, $L(u), L(v)$ are in F and thus $L'(t)$ also is in F.

- $L(S)$ may not be S_1 or S_3 since, by Lemma 23, it would have at least 3 arguments.

- If $L(S) = S_2$ then, by Lemma 25, $L(t) = \phi(w)$ for some $w \in \lambda_0 x.v$ and some $v \in F$. Then, choosing L' in such a way that $L'(t) = \phi(w_1)$ for $w_1 \in \lambda_1 x.v$ will give the desired properties .

(b) $t_1 = (S\ (K\ u)$, $t_2 = (K\ v)$ and $t' = (K\ (u\ v))$. Then $L(S)$ must be S_0 because otherwise, by Lemma 23, S would have at least two arguments. Similarly, we must have $L(K) = K_0$. Then, by Lemma 25, u, v are fair and thus setting $L'(S) = S_2$ and $L'(K) = K_0$ gives the desired properties.

(c) $t_1 = (S\ w_1)$ for $w_1 \in \lambda x.u_1$, $t_2 \in \lambda x.u_2$ and $t' \in \lambda x.v$ where v is a reduct of $(u_1\ u_2)$. Again by Lemma 23, we must have $L(S) = S_0$. By Lemma 25, $L(w_1) \in F$. By Lemma 39, extend L so that $L'(u_i) \in F$. Then setting L'' in such a way that $L''(t) \in \lambda_0 x.(u_1\ u_2)$ gives the desired properties.

Rule (4) : then $t = (S\ u_1\ u_2\ u_3)$, $L(S) = S_1$ and the $L(u_i)$ are fair. If $t' = (u_1\ u_3\ (u_2\ u_3))$ or if the reduced redex is in some u_i the result is trivial. Otherwise this means that, for $i = 1, 2$ $u_i \in \lambda x.v_i$ and $t' = (w\ u_3)$ for some $w \in \lambda x.v$ such that v is a reduct of $(v_1\ v_2)$. Then, by Lemma 39, extend L so that $L'(v_i) \in F$ and choose L'' in such a way that $L''(t) = (w'\ u_3)$ for $w' \in \lambda_1 x.(v_1\ v_2)$. □

Lemma 41 *Let t be a combinator. Assume that $t \triangleright v$ and $t \triangleright^* u$. Then, there is a labelling L of u and a term w such that $L(u)$ is fair, $L(u) \triangleright^* w$ and $v \triangleright^* \theta(w)$.*

Proof By induction on the length n of the reduction $t \triangleright^* u$.

- If $n = 1$, let L_0 be the labelling of t obtained by indexing all the occurrences of S, K and I by 0. $L_0(t)$ is clearly fair. Apply Lemma 40 to t, L_0 and the reduction $t \triangleright v$. This gives an extension L_1 of L_0. Applying Lemma 40 to t, L_1 and the reduction $t \triangleright u$ we get an extension L_2 of L_1. Applying the confluence of $\triangleright$ on fair terms (Theorem 36) to $L_2(t)$ gives the desired result.

- Otherwise, let $t \triangleright^* u_1 \triangleright u$. By the *IH*, let L_1 be a labelling of u_1 and w_1 be a term such that $L_1(u_1)$ is fair, $L_1(u_1) \triangleright^* w_1$ and $v \triangleright \theta(w_1)$. By Lemma 40, let L be a labelling of u_1 that is an extension of L such that $L(u_1)$ is fair and $L(u_1) \triangleright r$ for r such that $\theta(r) = u$. By theorem 36, let w be such that $r \triangleright^* w$ and $w_1 \triangleright^* w$. Then L, w have the desired properties.

□

Proposition 42 *The reduction given by rules (1, 2, 4) of definition 10 is confluent.*

Proof It is enough to show that, if $t \triangleright u$ and $t \triangleright^* v$ then $u \triangleright^* w$ and $v \triangleright^* w$ for some w. This follows immediately from Lemma 41. □

Definition 43 *I denote by $\supset$ the reduction defined by the following rules.*

1. $(S\ (K\ u)\ I) \supset u \qquad (K\ u\ v) \supset u \qquad (I\ u) \supset u$
2. $\lambda x.u \supset \lambda x.v$ *if* $u \supset v$

Lemma 44 *The reduction $\supset$ is confluent and commutes with $\triangleright$.*

Proof The reduction $\supset$ is strongly normalizing since it decreases the size. Thus to prove the confluence, it is thus enough to show the local confluence and this is straightforward. Since $\supset$ is also non duplicating, to prove the commutation with $\triangleright$, it is enough to show the local commutation and this is again straightforward. Note that the reductions $(K\ u\ v) \supset u$, $(I\ u) \supset u$ that are already present in $\triangleright$ are used here to ensure the confluence of the only critical pair i.e. $(S\ (K\ u)\ I\ w) \supset (u\ w)$ and $(S\ (K\ u)\ I\ w) \triangleright (K\ u\ w\ (I\ w))$.

□

Theorem 13 *The reduction given by rules (1, 2, 3, 4) of definition 10 is confluent.*

Proof Since $\rightarrow$ is the union of $\triangleright$ and $\supset$, the result follows immediately from proposition 42 and Lemma 44. □

4 Proof of theorem 7

Definition 45 *I denote by $\vdash$ the reduction defined by the following rules.*

1. $(S\ (K\ u)\ I) \vdash u \qquad (K\ u\ v) \vdash u \qquad (I\ u) \vdash u$
2. $[x]u \vdash [x]v$ *if* $u \vdash v$

Lemma 46 *The reduction $\vdash$ is confluent and commutes with $\succ$.*

Proof As in Lemma 44 □

Lemma 47 *If $u \rightarrow^* v$ then $u \succ^* w$, $v \vdash^* w$ for some w.*

Proof By induction on the length of the reduction $u \rightarrow^* v$. Assume $u \rightarrow u_1 \rightarrow^* v$. If the level of the reduction $u \rightarrow u_1$ is 0, the result follows immediately from the *IH* since then we also have $u \succ u_1$. Otherwise, the reduction looks like $u = C[\lambda x.t] \rightarrow u_1 = C[\lambda x.t_1] \rightarrow^* v$ where $t \rightarrow t_1$. By the *IH*, we have $t \succ^* w_1$, $t_1 \vdash^* w_1$ for some w_1 and thus $u \succ^* w_2$, $u_1 \vdash^* w_2$ where $w_2 = C[w_1]$. By the *IH* we also have $u_1 \succ^* w$, $v \vdash^* w$ for some w. By Lemma 46, we have $w_2 \succ^* w_3$ and $w \vdash^* w_3$ for some w_3 which is the term we are looking for. □

Theorem 7 *The reduction $\succ$ is confluent.*

Proof Assume $t \succ^* t_1$ and $t \succ^* t_2$. Then $t_1 \equiv t_2$ and thus, by theorem 12, $t_1 \approx t_2$. Since $\rightarrow$ is confluent we thus have $t_1 \rightarrow^* t_3$, $t_2 \rightarrow^* t_3$ for some t_3. By Lemma 47, let v_i be such that $t_i \succ^* v_i$ and $t_3 \vdash^* v_i$. Since $\vdash$ is confluent, let t_3 be such that $v_1 \vdash t_3$ and $v_2 \vdash t_3$. Since $\vdash$ is clearly a restriction of $\succ$, we have $t_i \succ^* t_3$ □

5 A standardization theorem

In this section I prove a standardization theorem for the system of section 3. I study this system instead of the one of section 2 because, as already mentioned in section 2.2, in the original system, what could be the leftmost redex is not clear at all.

Note that the following definition of a standard reduction does not need the definition of the residue of a redex. It is a definition by induction on $\langle lg(t \to t'), size(t)\rangle$ where $lg(t \to t')$ is the number of steps of the reduction. It uses the idea that is implicit in [1] and simply says that a standard reduction either reduces the head redex at the first step or is not allowed to reduce it.

Definition 48 *A reduction $t \to^* t'$ is standard ($t \to_{st} t'$ for short) if it satisfies the following properties.*

1. $t = (x\ \overrightarrow{u})$, $t' = (x\ \overrightarrow{u'})$ *and, for each* i, $u_i \to_{st} u'_i$
2. $t = (K\ u)$, $t' = (K\ u')$ *and* $u \to_{st} u'$.
3. $t = (S\ u)$, $t' = (S\ u')$ *and* $u \to_{st} u'$.
4. $t = (S\ u\ v)$ *and*
   - *either* $t' = (S\ u'\ v')$ *for* $u \to_{st} u'$ *and* $v \to_{st} v'$
   - *or the reduction is* $t \to t_1\ ...\ \to t_k \to_{st} t'$ *for some* $k \geq 0$ *such that* $t_i = (S\ u_i\ v_i)$, $u \to_{st} u_k$, $v \to_{st} v_k$ *and*
     - *either* $t_k = [x]w$, $t' = [x]w'$, $w \to_{st} w'$ *and, for each* $i < k$, t_i *cannot be written as* $[x]r$ *for some* r
     - *or* $u_k = (K\ u'_k), v_k = (K\ v'_k)$, *the reduction* $t_k \to_{st} t'$ *is* $t_k \to (K\ (u'_k\ v'_k)) \to_{st} t'$ *and, for each* $i < k$, t_i *cannot be written as* $(S\ (K\ u'_i)\ (K\ v'_i))$
     - *or* $u_k = (K\ u'_k), v_k = I$, *the reduction* $t_k \to_{st} t'$ *is* $t_k \to u'_k \to_{st} t'$ *and, for each* $i < k$, t_i *cannot be written as* $(S\ (K\ u'_i)\ I)$
5. $t = (I\ u_1\ ...\ u_n)$ *for* $n \geq 1$ *and*
   - *either* $t' = (I\ u'_1\ ...\ u'_n)$ *for* $u_i \to_{st} u'_i$
   - *or the reduction is* $t \to (u_1\ ...\ u_n) \to_{st} t'$
6. $t = (K\ u_1\ ...\ u_n)$ *for* $n \geq 2$ *and*
   - *either* $t' = (K\ u'_1\ ...\ u'_n)$ *for* $u_i \to_{st} u'_i$
   - *or the reduction is* $t \to (u_1\ u_3\ ...\ u_n) \to_{st} t'$
7. $t = (S\ u_1\ ...\ u_n)$ *for* $n \geq 3$ *and*
   - *either* $t' = (r\ u'_3\ ...\ u'_n)$ *where* $(S\ u_1\ u_2) \to_{st} r$ *and* $u_i \to_{st} u'_i$ *for* $i \geq 3$
   - *or the reduction is* $t \to (u_1\ u_3\ (u_2\ u_3)\ u_4\ ...\ u_n) \to_{st} t'$

Lemma 49
- *Assume* $u_i \to_{st} u'_i$ *for each* i. *Then* $(u_1\ ...\ u_n) \to_{st} (u'_1\ ...\ u'_n)$
- *Assume* $u \to_{st} [x]u'$. *Then* $(u\ v) \to_{st} u'[x := v]$

Proof Easy. □

Theorem 50 *If $t \to^* t'$ then $t \to_{st} t'$.*

Proof By induction on $lg(t \to^* t')$. It is enough to show that if $t \to_{st} t' \to t''$ then $t \to_{st} t''$. This is done by induction on $\langle lg(t \to_{st} t'), size(t)\rangle$ and by case analysis. We look at the rule that has been used to show $t \to_{st} t'$ and then what is the reduced redex in $t' \to t''$. I just consider two cases. The first one is typical and easy. The second one is similar but a bit more complex.

- $t = (K\ u_1\ ...\ u_n)$ for $n \geq 2$.
  - If the reduction is $t \to (u_1\ u_3\ ...\ u_n) \to_{st} t'$ we apply the *IH* to $(u_1\ u_3\ ...\ u_n) \to_{st} t' \to t''$ to get $(u_1\ u_3\ ...\ u_n) \to_{st} t''$ and thus $t \to (u_1\ u_3\ ...\ u_n) \to_{st} t''$ is standard.
  - If the reduction is such that $t' = (K\ u'_1\ ...\ u'_n)$ for $u_i \to_{st} u'_i$ then
    - either $t'' = (K\ u'_1\ ...\ u''_i\ ...\ u'_n)$ for $u'_i \to u''_i$ and we apply the *IH* to $u_i \to_{st} u'_i \to u''_i$ to get the result
    - or $t'' = (u'_1\ u'_3\ ...\ u'_n)$ and then $t \to (u_1\ u_3\ ...\ u_n) \to^* (u'_1\ u'_3\ ...\ u'_n)$ is standard by Lemma 49.
- $t = (S\ u_1\ ...\ u_n)$ for $n \geq 3$ and $t' = (r\ u'_3\ ...\ u'_n)$ where $(S\ u_1\ u_2) \to_{st} r$ and $u_i \to_{st} u'_i$ for $i \geq 3$. Assume also that $r = [x]a$, $x \notin r$ and $t'' = (a\ u'_4\ ...\ u'_n)$. This means that, for $i = 1, 2$, $u_i \to_{st} [x]v_i$ and that $(v_1\ v_2) \to_{st} a$. But then, by Lemma 49, $(u_i\ u_3) \to_{st} v_i[x := u_3]$. Thus, the following reduction is standard. $t \to (u_1\ u_3\ (u_2\ u_3)\ u_4\ ...\ u_n) \to_{st} (v_1[x := u_3]\ v_2[x := u_3]\ u_4\ ...\ u_n) \to_{st} (a[x := u_3]\ u_4\ ...\ u_n) \to_{st} (a\ u'_4\ ...\ u'_n) = t''$.

$\square$

6 Strong normalization of the typed calculus

In this section I prove the strong normalization of the auxiliary system of section 3. Note that the system of section 2 is not strongly normalizing even though this is for the following bad reason. Let $t = (S\ x\ x)$. Then $t = [y](S\ x\ x\ y) \succ [y](x\ y\ (x\ y)) = t$.

The types are the simple types i.e. constructed from basic types with the arrow. The typing rules are the usual ones i.e. I has type $A \to A$, K has type $A \to B \to C$, S has type $(A \to B \to C) \to (A \to B) \to A \to C$ for every types A, B, C and, finally, if u has type $A \to B$ and v has type A then $(u\ v)$ has type B.

Definition 51
- *A combinator t is highly normalizing ($t \in HN$ for short) if it can be obtained by the following rules.*
  1. *$t = S$ or $t = K$ or $t = I$ or $t = (x\ t_1\ ...\ t_n)$ for $t_1,\ ...,\ t_n \in HN$.*
  2. *$t = (K\ t_1)$ or $t = (S\ t_1)$ for $t_1 \in HN$*
  3. *$t = (S\ t_1\ t_2)$ for $(t_1\ x\ (t_2\ x)) \in HN$ where x is a variable.*
  4. *$t = (I\ t_1\ ...\ t_n)$ for $n \geq 1$ and $(t_1\ t_2\ ...\ t_n) \in HN$*
  5. *$t = (K\ t_1\ ...\ t_n)$ for $n \geq 2$, $(t_1\ t_3\ ...\ t_n) \in HN$ and $t_2 \in HN$*
  6. *$t = (S\ t_1\ ...\ t_n)$ for $n \geq 3$ and $(t_1\ t_3\ (t_2\ t_3)\ t_4\ ...\ t_n) \in HN$*
- *If $t \in HN$ we denote by $\eta(t)$ the number of rules that have been used to show $t \in HN$.*

We have introduced this notion of normalization which is stronger than the usual one (see the next Lemma) because the proof of Lemma 53 below would not work if HN was replaced by SN.

Lemma 52 *If $t \in HN$ then t is strongly normalizing.*

Proof By induction on $\eta(t)$. The non trivial cases are when the last rule that has been applied to prove $t \in HN$ is (3) or (6).

- Assume first $t = (S\ t_1\ t_2)$. Then, by the *IH*, $t' = (t_1\ x\ (t_2\ x)) \in SN$ and thus $t_1, t_2 \in SN$. Thus an infinite reduction of t must look like $t \to^* t'' \to^* ...$ where for some v_i, $t_i \to^* \lambda x.v_i$ and

  - either the reduction of t'' is in $(v_1\ v_2)$. But $(v_1\ v_2) \in SN$ since $t' \in SN$ and $t' \to^* (\lambda x.v_1\ x\ (\lambda x.v_2\ x)) \to^* (v_1\ v_2)$. Contradiction.

  - or $\lambda x.v_i = (K\ v_i)$ and the reduction is $t'' = (S\ (K\ v_1)\ (K\ v_2) \to (K(v_1\ v_2)) \to^*$ ... This is impossible since $t' \in SN$ and $t' \to^* (v_1\ v_2)$.

  -or $\lambda x.v_1 = (K\ v_1)$, $\lambda x.v_2 = I$ and the reduction is $t'' = (S\ (K\ v_1)\ I \to v_1 \to^*$ ... This is impossible since $t' \in SN$ and $t' \to^* v_1$.

- $t = (S\ t_1\ ...\ t_n)$. Again, by the *IH*, $t' = (t_1\ t_3\ (t_2\ t_3)\ t_4\ ...\ t_n) \in SN$. Thus the t_i are in SN and also $(S\ t_1\ t_2) \in SN$. The first point is clear. For the second, we argue as follows. Reasoning as in the previous case, it is enough to show that $(t_1\ x\ (t_2\ x)) \in SN$. If it was not the case then $(t_1\ t_3\ (t_2\ t_3)$ would also not been in SN and this contradicts the fact that $t' \in SN$. Thus an infinite reduction of t looks like $t \to^* (r\ t'_3\ ...\ t'_n) \to t'' \to^* ...$ where r is a reduct of $(S\ t_1\ t_2)$ and t'' is obtained by an interaction between r and its arguments. But we have shown (in the proof of theorem 50) that then t' reduces to t" and this is a contradiction.

□

Lemma 53 *Let t be a combinator and σ be a substitution such that all the variables in the domain of σ have the same type. Assume $t \in HN$ and the image of σ is included in HN. Then $\sigma(t) \in HN$.*

Proof By induction on $\langle type(\sigma), \eta(t)\rangle$. Look at the last rule that has been used to prove $t \in HN$. The only non trivial case is when $t = (x\ t_1\ ...\ t_n)$ and $x \in dom(\sigma)$. By the *IH*, $u_i = \sigma(t_i) \in HN$. We now have to distinguish the different possible values for $\sigma(x)$. The most difficult case (the other ones are similar or trivial) is when $\sigma(x) = (S\ a_1\ a_2)$. We have to show that $t' = (a_1\ u_1\ (a_2\ u_1)\ u_2\ ...\ u_n) \in HN$. But $t' = \tau((z\ u_2\ ...\ u_n))$ where z is a fresh variable such that $\tau(z) = (a_1\ u_1\ (a_2\ u_1))$. But $type(z) < type(x)$ and, by the *IH*, it is thus enough to show that $t'' = (a_1\ u_1\ (a_2\ u_1)) \in HN$. But $t'' = \tau'((a_1\ z'\ (a_2\ z')))$ where z' is a fresh variable such that $\tau'(z') = u_1$. Since $type(z') < type(x)$ and $(a_1\ z'\ (a_2\ z')) \in HN$ (because $(S\ a_1\ a_2) \in HN$), the result follows from the *IH*. □

Corollary 54 *Every typed combinator t is in HN and thus in SN.*

Proof By induction on the size of t using $(u\ v) = (x\ v)[x := u]$ and Lemma 53. □

7 Final remarks

Though intuitively quite simple, the given proof of confluence is technically rather involved and, in particular, it is more elaborate than the one using the confluence of the λ-calculus. Thus, one may wonder about the real use of such a proof even if this is the condition to have a self contained theory. I will argue for another reason.

Combinatory Logic somehow looks like a calculus with explicit substitutions. Though $([x]u\ v)$ is not exactly the *explicit* substitution $u[x := v]$, it has often to be understood in this way. In particular, the reduction $([x](u_1\ u_2)\ v) \to ([x]u_1\ v\ ([x]u_2\ v))$

looks like the propagation of the substitution into the two branches of the application. But proving confluence for such calculi is usually not trivial simply because the usual methods (parallel reductions or finite developments) need definitions that are not clear.

I thus hope that the given proof will help in finding simple proofs for calculi with explicit substitutions.

Acknowledgments

I wish to thank R. Hindley and the anonymous referee for helpful comments on previous versions of this paper.

Added in proof

Shortly after sending the first version of this paper, I have been informed by R.Hindley and P.Minari that the later has also written (more or less at the same time) a direct proof of the confluence of combinatory strong reduction. This proof is completely different from the one given here. See the TLCA list of open problem or [7].

References

[1] R. David, *Une preuve simple de résultats classiques en λ-calcul* C. R. Acad. Sci. Paris, t.320, Srie I, 1995, pp 1401 -1406.

[2] H.B. Curry & R. Feys, *Combinatory Logic, Volume I.* North Holland (3rd edition 1974).

[3] H.B. Curry, J.R. Hindley & J.P.Seldin, *Combinatory Logic, Volume II.* North Holland.

[4] J.R Hindley, *Axioms for strong reduction in combinatory logic* Journal of symbolic logic 32-2, 1967, pp 237-239

[5] J.R. Hindley & J.P. Seldin, *Introduction to Combinators and λ-calculus*, Cambridge University Press 1986.

[6] B. Lercher *The decidability of Hindley's axioms for strong reduction*, Journal of symbolic logic 1967, 32-2, pp 224-236

[7] P. Minari *A solution to Curry and Hindleys problem on combinatory strong reduction*, to appear in Archive for Mathematical Logic vol. 48(2), 2009